

Improving the Performance of Remote I/O Using Asynchronous Primitives

Nawab Ali and Mario Lauria

Department of Computer Science and Engineering

The Ohio State University

Columbus, OH 43210

{alin, lauria}@cse.ohio-state.edu

- Introduction
- Storage Resource Broker
- SEMPLAR
 - Design
 - Implementation
- Remote Asynchronous I/O
 - Design & Implementation
- Experimental Setup
- Results
- Conclusion

- **Application Trends**

- Big Science projects increasingly require processing of large data sets
 - Sloan Digital Sky Survey, Large Hadron Collider, National Earthquake Engineering Simulation Grid, etc
- Large data sets stored in repositories at specialized facilities (supercomputer centers)

- **Technological Trends**

- Bandwidth of WAN and Internet backbones growing at a rate that makes it comparable to local interconnect speed
 - TeraGrid, LambdaRail ~40Gb/s
 - Infiniband ~ 10 Gb/s

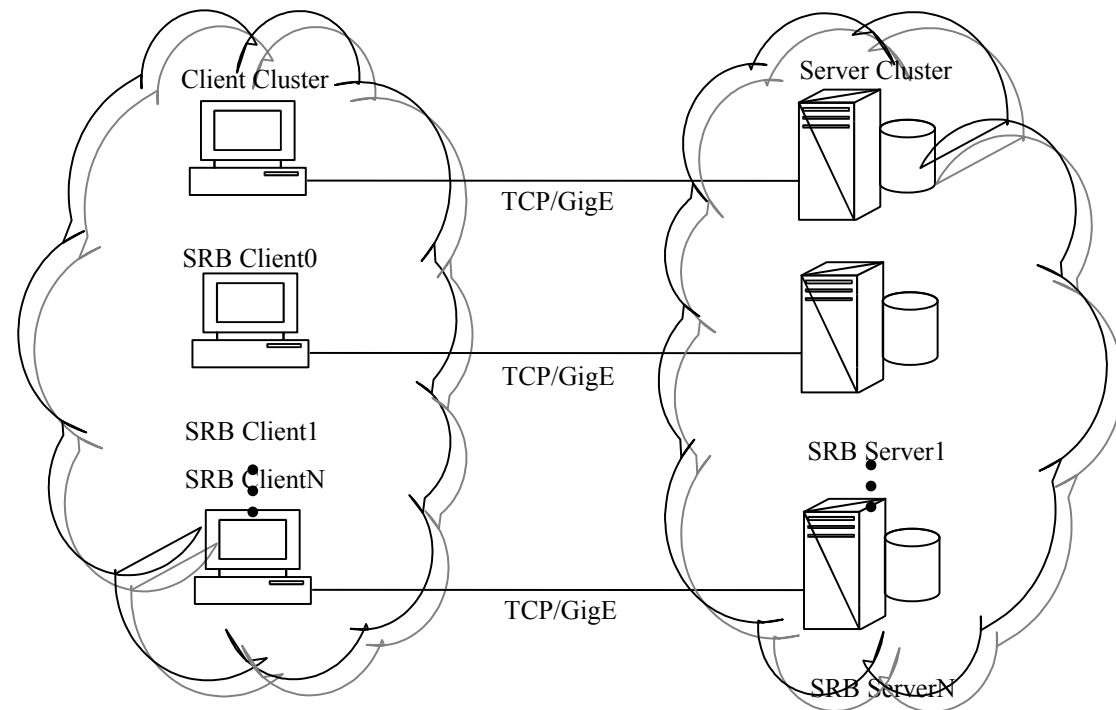
- Common approach to processing large data sets
 - Local staging vs. Direct remote access
- Problems with staging
 - Detailed knowledge about the remote filesystem
 - Overlapping of data transfer and computation not possible
 - Dynamic data sets require frequent refreshes of the local copy
- Previous attempts at building remote data access tools (RIO, GridFTP) based on incremental improvement of legacy tools haven't proven competitive with local staging

Storage Resource Broker

- SRB was developed at SDSC to provide remote I/O to massive volumes of data in a production environment
- It provides transparent access to heterogeneous storage resources
 - Filesystems
 - Database Systems
 - Archival Storage Systems
- Other services offered
 - Authentication, location transparency

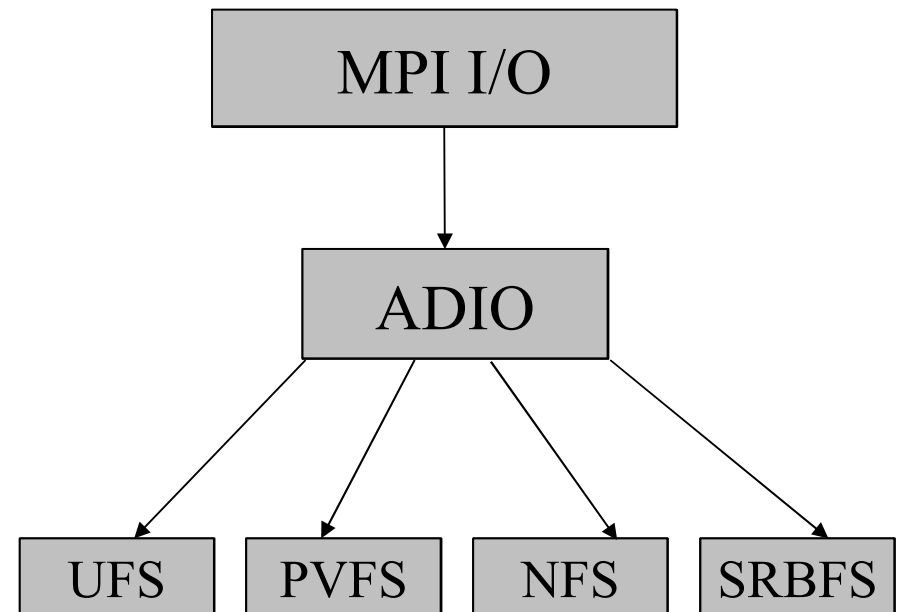
SEMPLAR: SRB Enabled MPI I/O Library for Access to Remote Storage

- I/O over the Internet
- Storage Virtualization
 - SRB
- High I/O bandwidth
 - Multiple TCP Streams
 - Multiple I/O nodes
- Standard Application Interface
 - MPI I/O



SEMPLAR Implementation

- MPI I/O implementations such as ROMIO use the portable ADIO interface
- ADIO implementations are optimized for a particular filesystem
- We have provided an ADIO implementation for the SRB filesystem



Remote Asynchronous I/O

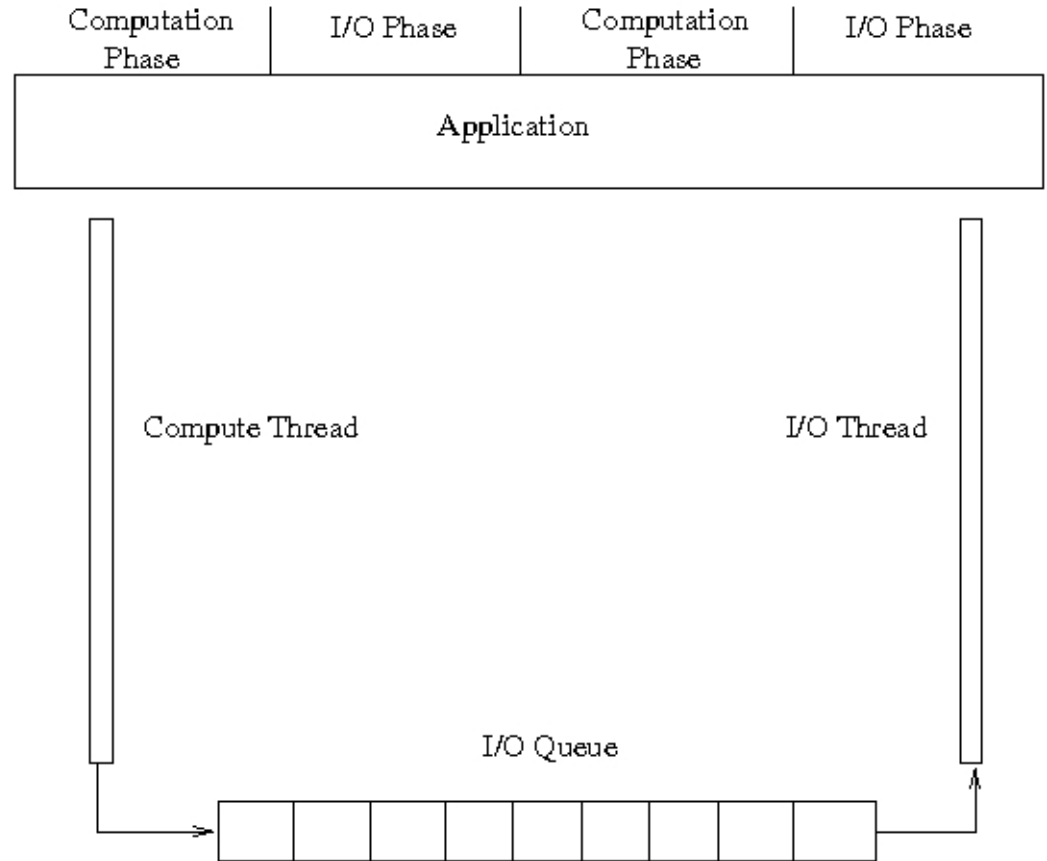
- Powerful programming model
- Enables applications to parallelize I/O and computation
- Increased CPU utilization
- Enhanced I/O performance
- Higher network throughput
 - Multiple concurrent TCP streams
 - On the fly data compression

Asynchronous I/O Implementation

- Multi-threaded
 - Compute thread
 - I/O thread
- Shared I/O queue
- Asynchronous calls place I/O requests in queue and return immediately
- I/O thread dequeues I/O queue in FIFO order

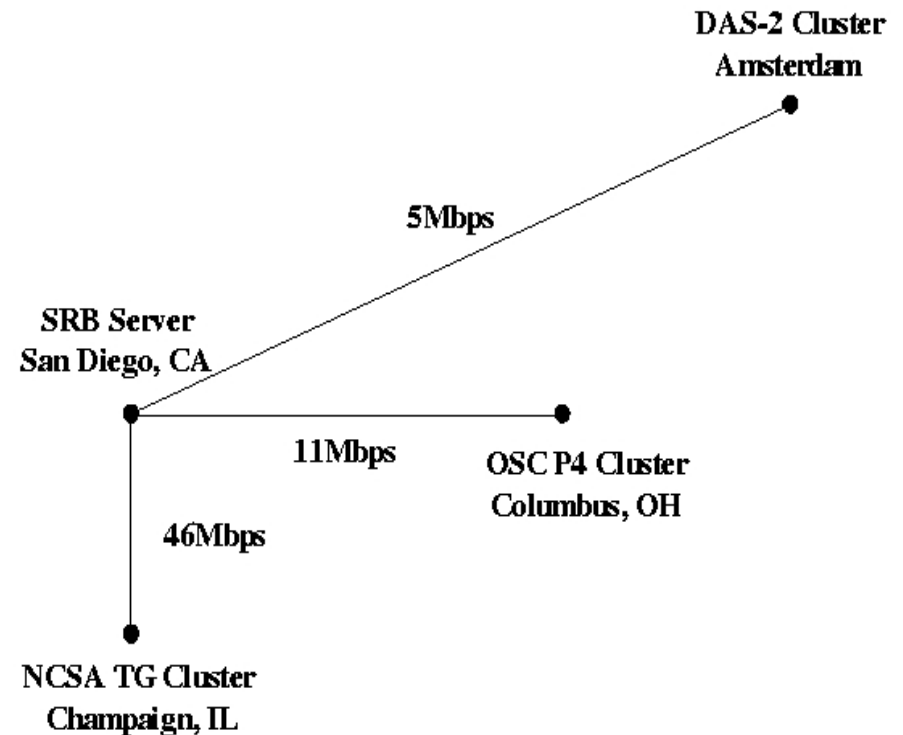
Asynchronous I/O Implementation

- POSIX pthread library
- Asynchronous Primitives
- ***MPI_File_iread***
- ***MPI_File_iwrite***
- ***MPIO_Wait***
- ***MPIO_Test***



Experimental Setup

- SRB server v3.2.1 running on orion.sdsc.edu
- NCSA TeraGrid cluster
 - High bandwidth, Low latency
- DAS -2
 - Low bandwidth, High Latency
- Intel Pentium 4 Xeon cluster at OSC
 - High bandwidth, Low latency
 - Private I/P addresses



- ROMIO perf
 - Measures the read and write performance
 - Synchronous and Contiguous I/O
 - Upper-bound on the MPI I/O performance
- 2D Laplace Solver
 - Solves Laplace's equation on a 2D grid
 - Writes a checkpoint file periodically
- MPI-BLAST
 - Searches protein and nucleotide databases for local alignment
 - MPI wrapper for BLAST

2D Laplace Pseudocode

```

program 2D_LAPLACE_SYNCHRONOUS_IO
BEGIN
  for i ← 1...N
  DO
    <Compute the Overall Residual>
    <Send Phase>
    <Receive Phase>
    <Checkpoint Data using Synchronous I/O>
  DONE
END

```

```

program 2D_LAPLACE_ASYNCHRONOUS_IO
BEGIN
  for i ← 1...N
  DO
    <Compute the Overall Residual>
    2 → <Send Phase>
        <Receive Phase>
    1 → <Wait for Previous I/O to Complete>
        <Checkpoint Data using Asynchronous I/O>
  DONE
  <Wait for Last I/O to Complete>
END

```

MPI-BLAST pseudocode

```

program MPI_BLAST_SYNCHRONOUS_IO
BEGIN
  LOOP
  DO
    <Request Master for new Sequence>
    IF <Sequence == NULL> THEN
      BREAK
    ELSE
      <Run BLAST on Sequence>
      <Write Output using Synchronous I/O>
    DONE
  END

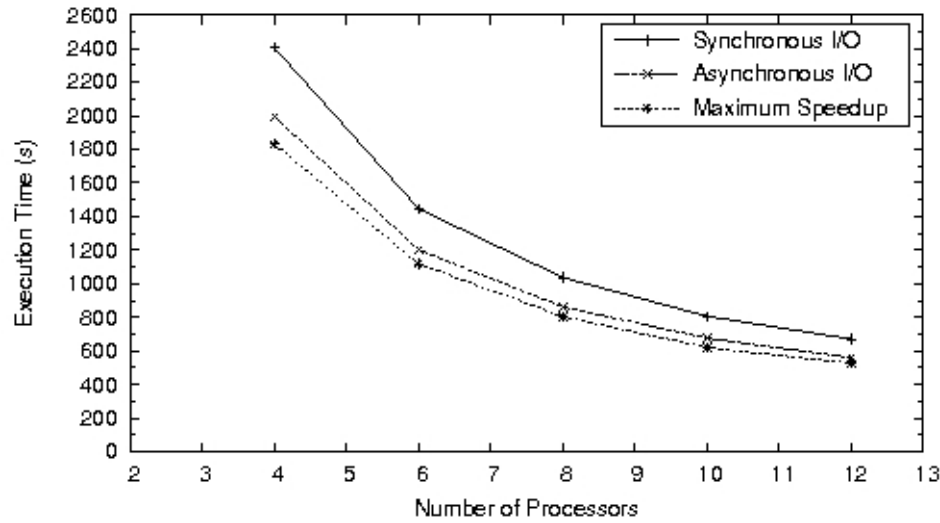
```

```

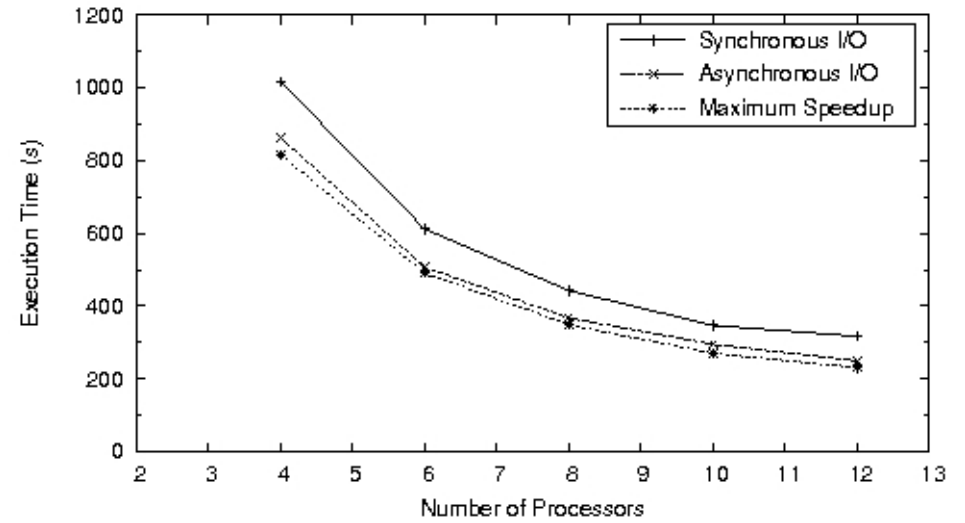
program MPI_BLAST_ASYNCHRONOUS_IO
BEGIN
  LOOP
  DO
    <Request Master for new Sequence>
    IF <Sequence == NULL> THEN
      BREAK
    ELSE
      <Run BLAST on Sequence>
      <Wait for Previous I/O to Complete>
      <Write Output using Asynchronous I/O>
    DONE
    <Wait for Last I/O to Complete>
  END

```

MPI-BLAST I/O Performance

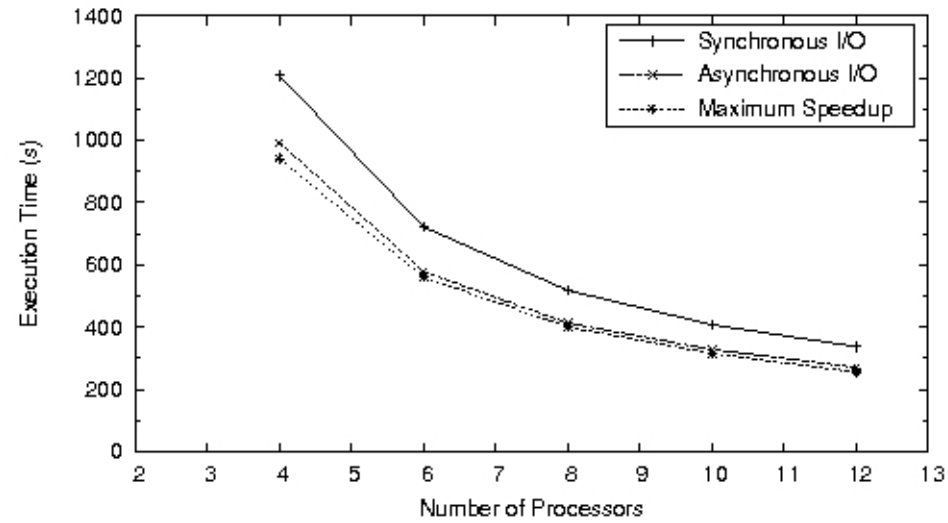


DAS-2 Cluster



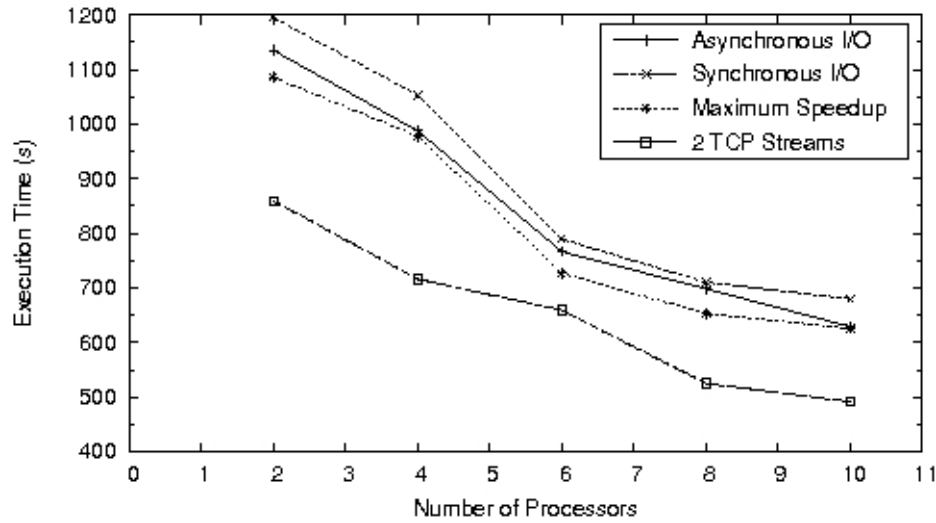
NCSA TeraGrid Cluster

MPI-BLAST I/O Performance

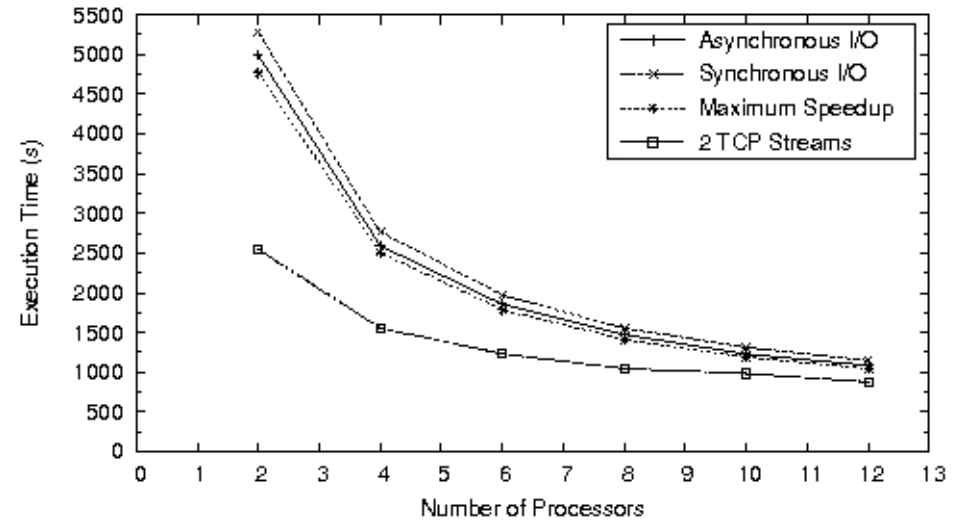


OSC P4 Cluster

2D Laplace Solver

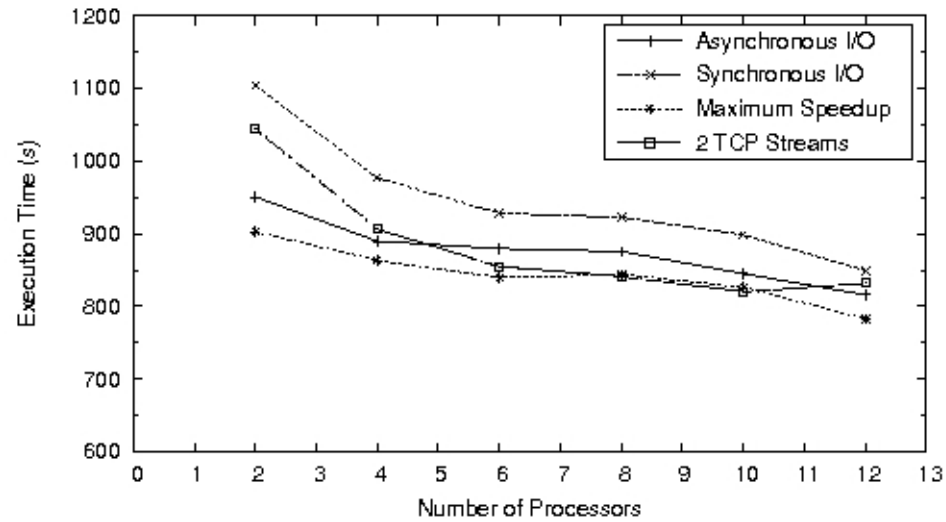


NCSA TeraGrid Cluster



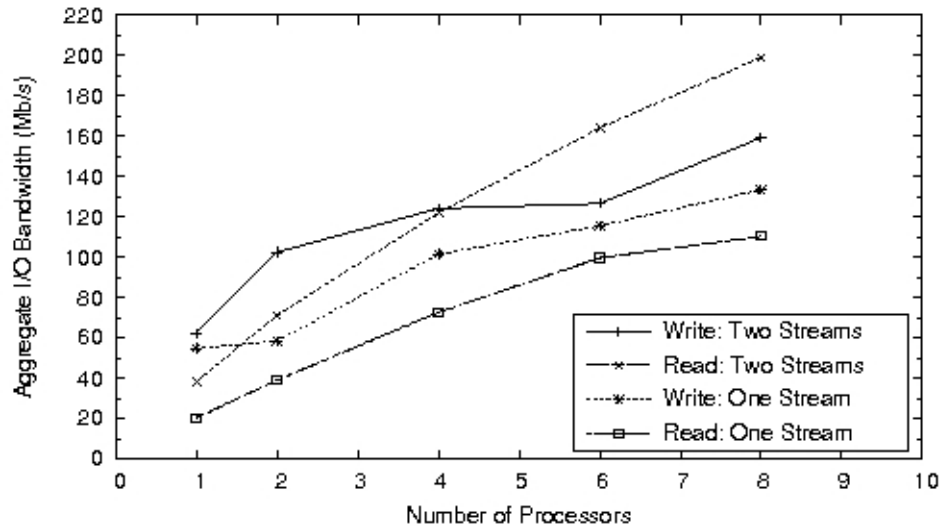
DAS-2 Cluster

2D Laplace Solver

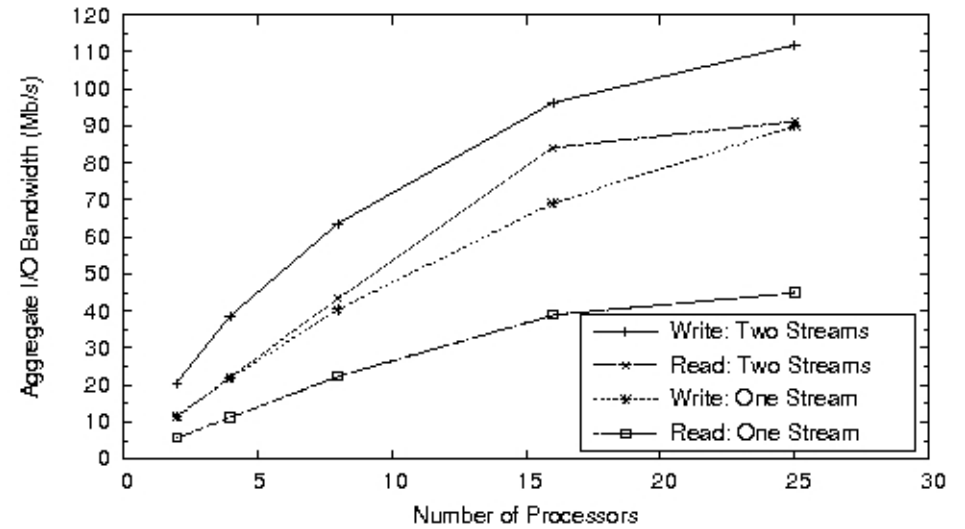


OSC P4 Cluster

Perf I/O Performance

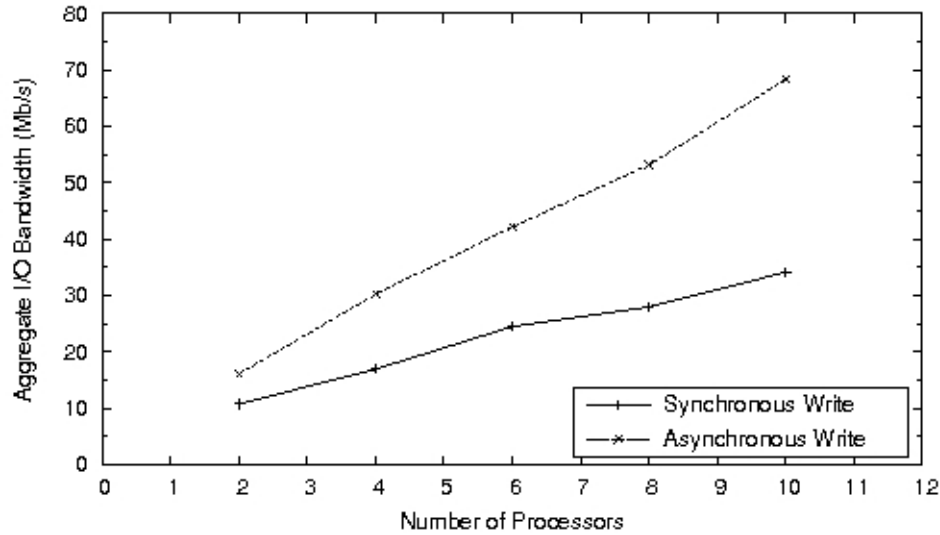


NCSA TeraGrid Cluster

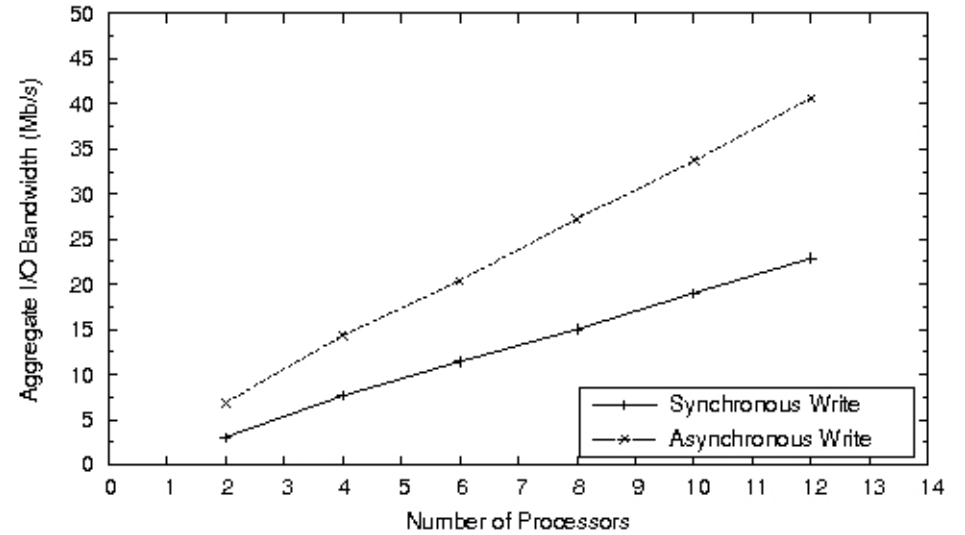


DAS-2 Cluster

On-the-fly Data Compression



NCSA TeraGrid Cluster



DAS-2 Cluster

- **MTIO**
 - Multi-threaded MPI based I/O library [More et al.]
- **RFS**
 - Active Buffering with threads (ABT)
- **AMPED**
 - Asymmetric, multi-process, event-driven
- **GASS**
 - Caching Schemes for common grid file access patterns

- High-Performance remote I/O is an active area of research
- Asynchronous primitives improve the performance of remote I/O
 - Overlap computation with I/O
 - Asynchronous Split-TCP
 - On-the-fly data compression
- Present the results of a high-performance computing workload on 3 clusters

- Data redundancy across multiple concurrent I/O streams
- Study the effect of asynchronous I/O on remote, collective I/O
- Dynamic degree of parallelism
 - Adjust the number of connections based on the network load

Thank You